REMARKS

In the Non-Final Office Action dated February 11, 2004, claims 1-20 are pending. Claims 1, 8, and 11 are independent claims from which all other claims depend therefrom. Claims 1, 7-8, 10-11, 16, and 18-20 have been amended.

Claims 11 and 20 stand objected for informality reasons. Claim 11 stands objected to because line 2 recites "of one or more" where it should recite "one or more". Claim 11 has been amended to remove "of" from "of one or more". Claim 20 stands objected to because it is not numbered. Claim 20 has been amended to be correctly numbered as claim 20.

Claims 1-7 and 16-19 stand rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to point out and distinctly claim the subject matter which applicant regards as the invention. Specifically, the Office Action states that the term "referring to" in line 8 of claim 1 renders claim 1 indefinite and also state that it is not clear what elements are referring to the state table and how the state table is used by the elements. The Office Action further states that it is not clear if "referring to" includes merely using information derived from the state table. Claim 1 has been amended to remove the term "referring to" and to clarify how the state table is used. The currently amended claim 1 recites the implementation of a state model by running a runtime code while utilizing information within a state table. Applicants submit that this recited limitation or element of claim 1 utilizes information within the state table in implementing the state model. Note that claim 1 is a method claim and that in being as such no device or component need be recited in performing a method step or element. Nevertheless, as an example, a controller may utilize the information within the state table when running the runtime code. For example, as stated in paragraph [0028] of the specification of the present application a real time controller may use state table information at run-time to perform state transitions and to identify the actions and conditions associated with the

transitions. Additional examples are provided in paragraph [0032] of the specification of the present application.

With regards to claims 7 and 19, the Office Action states that it is unclear how a state model can be "annotated" with "actions and conditions". Claims 7 and 19 have been amended to clarify what is meant by the term "annotating". Claim 7 now recites annotating the state model using a script programming language to alter state behavior. Claim 19 now recites a scripted dynamic events processor for annotating one or more state models to alter state behavior. The term "annotating" refers to the altering of state models using a script language, such as Tcl, to change state behavior without having to rebuild a runtime code, as stated in paragraph [0021] of the specification of the present application. The altering of state models may include the embedding of additional language or extension language. The term "annotating" does not refer simply to the adding of comments or notes, but rather to the adding of extension language that alters state behavior during run-time. Note that it is well settled that a patentee may define a claim term either in the written description of the patent or in the prosecution history, as in this case, *Mycogen Plant Science v. Monsanto Co.*, 243 F.3d 1316, 1327, 58 USPQ2d 1030, 1039 (Fed.Cir. 2001).

Regarding claim 16-19, the Office Action states that it is not clear how to interpret the "runtime library". The Office Action further states an accepted meaning of the term "library" is a collection of subroutines and functions stored in one or more files, and states that it is unclear how the runtime library of claims 16-19 are able to perform functions. Claims 16 and 18-19 have been amended to clarify what is meant by a "runtime library". Claim 16 now recites that the runtime library comprises an event processor and an interpreter, which are capable of performing functions. Also Applicants, respectfully refer the Examiner to paragraph [0022] of the specification, which provides a runtime library and sample devices contained therein for performing functions. Some of

the devices that may be contained within the runtime library are recited in the limitations of claims 17-19, which are dependent on claim 16. Applicants submit that the runtime library has been clearly defined in the specification and is not simply a collection of subroutines and functions stored in one or more files, but rather includes one or more devices for the performance of various functions.

Thus, Applicants submit that the objections and rejections under 35 U.S.C. 112 have been overcome and that claims 1-7, 11, and 16-20 are now in a condition for allowance at least with respect thereto.

Claims 1-6, 8, 11-15, and 20 stand rejected under 35 U.S.C. 102(a) as being anticipated by Russell (USPN 6,212,625 B1).

Claim 1 recites a method of implementing a pre-designed state model. The method includes: A.) extracting state information from the state model; B.) processing the extracted state information; C.) generating a state code and a state table in response to the processed extracted state information; D.) compiling the state code to generate a runtime code; and then E.) implementing the state model by running the runtime code while utilizing information within the state table using a separate controller.

The Office Action states that Russell teaches the limitations of claim 1 and in doing so refers to col. 2, lines 40-48. Applicants submit that in col. 2, lines 40-48, Russell discloses a traditional approach that includes: I.) loading a state table file or hardware description file; II.) extracting state information for the executing of a finite state machine from the state table file; and III.) generating a data structure from the extracted information. The data structure represents the finite state machine. Thus, in col. 2, lines 40-48, Russell discloses the generation of a finite state machine representation from a hardware description file. Russell does not disclose any of elements A-E recited above.

The method of claim 1 generates a state code and a state table in response to extracted information from a pre-designed state model or state machine representation. This is different than the traditional approach stated in Russell, which generates a finite state machine representation from a state table or a hardware description file. The method of claim 1 begins with a state representation and ends with the running of a runtime code utilizing information within a state table, whereas the approach provided in Russell begins with a state table and ends with a finite state machine representation.

Not only are the starting points and ending results of claim 1 and the traditional approach of Russell different, the elements or tasks performed therebetween are also different. For example, the approach of Russell does not generate a state code or a state table and does not compile the state code to generate a runtime code, but rather simply extracts state information from a state table file to generate a data structure.

Additionally, the state table file of the traditional approach of Russell does not appear to be similar to the state table of claim 1. Although it is stated that the state table file of Russell contains information for the execution of a finite state machine, the information is used for the generation of a data structure that represents the finite state machine. The state table file and the information contained therein of Russell is not used during the actual execution of the finite state machine nor is the information in the state table file extracted from a state model or representation of the finite state machine, as is the state table of claim 1. The state table of claim 1 contains information to be used during run-time to perform state transitions and to identify the actions and conditions associated with those transitions. Thus, the state table file in the traditional approach of Russell is not the same as the state table of claim 1.

Russell also discloses a state engine for the execution of finite state machines. Russell discloses the use of an input and filter unit 300, a transition

unit 302, a storage unit 304, and an action generation unit 306, which are used in the execution of the finite state machines. The state engine of Russell executes state machines sequentially. In Russell a single state machine is loaded and entries regarding that state machine are loaded into a state entry table 510. Upon completing the execution of a state machine, entries for a next state machine may be loaded. In Russell entries for a particular state machine are loaded into the state table 510, inputs are received by the filter 300 and converted into symbols, the transition unit 302 compares state entries in the state entry table 510 with a current state and a current symbol to determine terminating entries to perform, which are then executed by the action generation unit 306.

The engine of Russell does not include a device for the extraction of state model information from a state model, a device for the separation of state model information into a state code and a state table, and a device for the execution of a state code using information within a state table. The loading of symbols, by Russell, into a state table for the execution of particular functions is not the same as the separation of state model information into a state code and a state table, and the utilization of the state table during the execution of a runtime code. The state table 510 of Russell is utilized before the execution of terminating entries and is used to determine entries to execute whereas the state table of claim 1 is utilized during the execution of a runtime code to perform state transitions.

Furthermore, unlike Russell, the state table utilization of claim 1 does not affect the hardware specific functionality or basic functions of a controller. The tasks performed by the engine of Russell are performed within and executed by a single controller. Elements A-D of claim 1 are performed separate from element E, as is denoted by the use of a separate controller. Thus, each and every element of claim 1 is not taught or suggested by Russell, therefore claim 1 is novel, nonobvious, and is in a condition for allowance.

Claim 8 recites a method for implementing a pre-designed plurality of state models for a state machine having an event configuration file. The method includes: i.) extracting state information from the state models; ii.) generating an events symbols header, having global and shared event symbol definitions, from the event configuration file; iii.) processing the extracted state information in response to the events symbols header; iv.) generating state codes and state tables in response to the processed extracted state information; v.) compiling the state codes using the events symbols header to generate runtime codes; and vi.) implementing the state models by running the runtime codes while referring to the state tables.

As stated above, Russell does not teach or suggest the above elements i, iv, and vi. In regards to elements ii, iii, and v, the Office Action states that elements ii, iii, and v are taught by Russell and refers to col. 5, lines 37-59. Applicants, respectfully, traverse and submit that Russell fails to teach or suggest the use of an events symbol header as defined by the method of claim 8. Russell in col. 5, lines 37-59 discloses the use of the state entry table 510 having a state identifier, a symbol identifier, multiple state attributes, and a next state, none of which being an events symbol header. The events symbol header of claim 8 contains global and shared event symbol definitions. The events symbol header contains a centralized list of events for easy addition or renaming of events and for the prevention of duplicating of stored events, as stated in paragraph [0020] of the specification of the present application. All the above stated entries contained within the state entry table 510 of Russell correspond at any given time with a single finite state machine, whereas the events symbol header of claim 8 corresponds with multiple state models. Thus, each and every element of claim 8 is also not taught or suggested by Russell, therefore claim 8 is also novel, nonobvious, and is in a condition for allowance.

PAGE 12/14 * RCVD AT 3/9/2004 11:46:00 AM [Eastern Standard Time] * SVR:USPTO-EFXRF-1/0 * DNIS:8729306 * CSID:248 2239522 * DURATION (mm-ss):03-26

Claim 11 recites a state processor for generating a state table and a runtime code for use in the implementation of one or more pre-designed state models. The processor of claim 11 includes a state model information provider that extracts state model information in response to the state models. A state information separator generates a state code and the state table in response to the one or more state models. A compiler compiles the state code and generates the runtime code. As stated above, Russell does not teach or suggest the generation of a state code and the state table in response to one or more state models. Thus, Russell also does not teach or suggest the state information separator of claim 11. Since Russell does not teach or suggest each and every element of claim 11, claim 11 is also novel, nonobvious, and is in a condition for allowance.

Applicants further submit that since claims 2-6, 12-15, and 20 depend from claims 1, 8, and 11, respectively, claims 2-6, 12-15, and 20 are also novel, nonobvious, and are in a condition for allowance for at least the same reasons as put forth above with respect to claims 1, 8, and 11.

Claim 7 stands rejected under 35 U.S.C. 103(a) as being unpatentable over Russell. Claims 9-10 and 16-19 stand rejected under 35 U.S.C. as being unpatentable over Russell in view of Bernaden III et al. (USPN 6,477,439 B1). Applicants submit that since claims 7, 9-10, and 16-19 depend from claims 1, 8, and 11, that claims 7, 9-10, and 16-19 are also novel, nonobvious, and are in a condition for allowance for at least the same reasons as put forth above with respect to claims 1, 8, and 11.

In light of the amendments and remarks, the Applicants submit that all objections and rejections are now overcome. The Applicants have added no new matter to the application by these amendments. The application is now in condition for allowance and expeditious notice thereof is earnestly solicited. Should the Examiner have any questions or comments, he is respectfully requested to call the undersigned attorney.

Respectfully submitted,

ARTZ & ARTZ P.C.

Jeffrey J. Chapp, Reg. No. 50,579
28333 Telegraph Road, Suite 250
Southfield, MI 48034
(248) 223-9500

Dated: March 9, 2004